ROBUST IP AND UDP-LITE HEADER RECOVERY FOR PACKETIZED MULTIMEDIA TRANSMISSION

François Mériaux[1,2]*, Michel Kieffer*[2]

[1]Département EEA Ecole Normale Supérieure de Cachan
[2]L2S - CNRS - SUPELEC - Univ Paris-Sud

ABSTRACT

Recently, Joint Source-Channel Decoding (JSCD) techniques have been proposed to improve the reception of multimedia contents transmitted over error-prone channels. These techniques take advantage of the redundancy left by the source coder and of bit reliability measures (soft information) provided by channel decoders to correct transmission errors. To be put at work, protocol stacks have to be made permeable to transmission errors in order to allow soft information to reach the upper protocol layers. For that purpose, headers have to be reliably estimated at each protocol layer. First results have been obtained for lower protocol layers (PHY and MAC) protected by CRCs. The aim of this paper is to extend these results to upper protocol layers (IP and UDP-lite) protected by checksums. As for CRCs, trellis-based decoding techniques may be employed for data protected by checksums. Nevertheless, specific tools have been proposed in this paper to reach a complexity-efficiency trade-off[1].

Index Terms— Codes, Communication systems, Transport protocols, Redundancy, MAP estimation.

1. INTRODUCTION

The delivery of multimedia contents over a bandlimited network requires the use of compression schemes to reduce the amount of data to transmit. Nevertheless, compressed data are highly sensitive to transmission errors: a single bit corrupted during its transmission may have heavy consequences on the full encoded content. To avoid such issue, contents which have to be transmitted through a network are packetized and protected with various error-correction and detection mechanisms. The basic protection is channel coding [1], which adds redundancy to the information as it is sent over the physical channel. Another protection mechanism is included in the network architecture and its different layers protocols. For some of those layers, an error-detection code is added to the message to protect the header of the packet. These error-detection codes can be CRCs or checksums [1, 4]. At the receiver, these codes are used to decide whether the message is corrupted and if it is, the sender is asked to send again the packet, if possible.

Nevertheless, there are many live applications where packets cannot be sent again: broadcasting, voice over IP, etc. For the last few years, *Joint Source-Channel Decoding* techniques [3] have been proposed to deal with corrupted packets by correcting them using various sources of redundancy present in the whole coding and transmission chain. These techniques improve the decoding but as they treat the corrupted packets at the application layer, they require the network to be permeable to such erroneous packets. The current network architecture does not allow packets with corrupted headers to reach the application layer because of the error-detection codes.

In [**?**], a mechanism is proposed to allow the lower protocol layers (PHY and MAC) to be permeable to corrupted packets by using the CRCs in conjunction with inter- and intra-layer redundancy to correct the erroneous headers.

The main contribution of this paper is to extend the ideas in [**?**] to the IP and UDP-lite protocol layers. The headers of these layers are protected by checksums. As for CRCs, trellis-based decoding techniques [6] may be employed for data protected by checksums. Nevertheless, we propose specific tools have been proposed in this paper to reach a complexity-efficiency trade-off.

The paper is organized as follow. We first explain the header correction mechanism, then we introduce an efficient way to reduce its complexity. Finally, we present the simulation context and the results of our method.

2. ROBUST RECOVERY HEADERS AT THE HIGHER LAYERS USING THE CHECKSUMS

2.1. MAP estimator for robust header recovery

By keeping the same formalism than in [**?**], consider a packet forwarded from a layer L to layer $L+1$. This packet is composed of a payload and a header and in this header, some fields are protected by a checksum. We note $\mathbf{r}$ the vector containing the protected fields. Mny of these fields are *unknown* but as there are many sources of redundancy inter and intra-layer, some of them can ben *known* or *predictable*. We respectively note those vectors $\mathbf{u}$, $\mathbf{k}$ and $\mathbf{p}$ and we have $\mathbf{r} = [\mathbf{k}, \mathbf{p}, \mathbf{u}]$.

[1]This work has been partly supported by the ANR DITEMOI project.

We also define the checksum vector $\mathbf{c} = C_n(\mathbf{r})$, with C_n being the n-bit checksum evaluation function and we consider R, the implicit information at the receiver which enables us to limit the search range of unknown fields. The soft information coming from L are regrouped in the vector $\mathbf{y} = [\mathbf{y}_k, \mathbf{y}_p, \mathbf{y}_u, \mathbf{y}_c]$, $\mathbf{y}_k$,$\mathbf{y}_p$,$\mathbf{y}_u$ and $\mathbf{y}_c$ respectively being the noisy values of the *known*, *predictable*, *unknown* and checksum fields.

Since the fields $\mathbf{k}$ and $\mathbf{p}$ are known or exactly predictable, the only remaining field to be estimated is $\mathbf{u}$. This problem can be solved with a MAP estimator

$$\hat{\mathbf{u}}_{\mathrm{MAP}} = \arg\max_{\mathbf{u}} P(\mathbf{u} \mid \mathbf{k}, \mathbf{p}, \mathrm{R}, \mathbf{y}_u, \mathbf{y}_c) \tag{1}$$

After some derivations detailed in [**?**], one gets

$$\hat{\mathbf{u}}_{\mathrm{MAP}} = \arg\max_{\mathbf{u}} P(\mathbf{y}_u \mid \mathbf{u}) P(\mathbf{y}_c \mid C_n(\mathbf{k}, \mathbf{p}, \mathbf{u})) P(\mathbf{u} \mid \mathrm{R}) \tag{2}$$

2.2. Practical evaluation of the MAP estimator

In (2), the first term $P(\mathbf{y}_u \mid \mathbf{u})$ represents the likelihood of the sequence $\mathbf{u}$ when having the soft information $\mathbf{y}_u$. The second term $P(\mathbf{y}_c \mid C(\mathbf{k}, \mathbf{p}, \mathbf{u}))$ represents the likelihood of the checksum of the sequence $\mathbf{r} = [\mathbf{k}, \mathbf{p}, \mathbf{u}]$. The last term $P(\mathbf{u} \mid \mathrm{R})$ may limits the range of possible values for $\mathbf{u}$.

If we consider $l(\mathbf{u})$, the length in bits of the unknown vector $\mathbf{u}$, $2^{l(\mathbf{u})}$ possible combinations for $\mathbf{u}$ have to be considered. But, as $l(\mathbf{u})$ can be quite long, we cannot afford such a calculation complexity. That is why we use a trellis structure similar to the one described in [6] to compute $P(\mathbf{y}_u \mid \mathbf{u})$. In our case, the rows of the trellis correspond to the different possible checksum values and each column of the trellis corresponds to one bit of the unknown sequence. Each bit of the unknown sequence offers two new possible checksum values to the existing ways. At the end of the trellis, we have the likelihood metrics of $2^{l(\mathbf{c})}$ different possible sequences and their associated checksums.

To compute $P(\mathbf{y}_c \mid C(\mathbf{k}, \mathbf{p}, \mathbf{u}))$, we just have to compare those checksums with $\mathbf{y}_c$ by the mean of a likelihood metric.

3. COMPLEXITY REDUCTION

Both IP and UDP checksums are coded over 16 bits, which means we have to work with a 2^{16} rows trellis. But such a trellis requires a huge amount of calculation and we need to reduce its complexity. The solution proposed in [**?**] cannot be applied in our case as we cannot divide the checksum into several independent pieces.

3.1. From a $2n$-bit checksum to a n-bit checksum

As detailed in [2], for a given sequence it is possible to obtain its n-bit checksum from its $2n$-bit checksum. Let us consider K $2n$-bit words W_k, $k \in [1, K]$. One can divide each W_k word into two n-bit words such as the first one is made of the n most significant bits from W_k and the second one is made of the n least significant bits from W_k. We respectively note those words W_k^m and W_k^l and $W_k = W_k^m \times 2^n + W_k^l$. One can write

$$C_n(C_{2n}(W_1, ..., W_K)) = C_n(W_1^m, W_1^l, ..., W_K^m, W_K^l) \tag{3}$$

Practically, once we have the $2n$-bit checksum $\mathbf{c}_{2n}$ of a given sequence, we just do the 1's complement sum of of the n most significant bits of $\mathbf{c}_{2n}$ with the n least significant bits of $\mathbf{c}_{2n}$ to obtain the n-bit checksum $\mathbf{c}_n$ of the same sequence. This property enables us to work with a n-bit checksum rather than a $2n$-bit one so the complexity of the sequence estimation is highly reduced. But the counterpart is that a n-bit checksum is a worse protection than a $2n$-bit one. The results of errors correction with this method are not as good as with the $2n$-bits checksum, but we will see this in detail in section 4.2.

3.2. Computing the likelihood of the n-bit checksum

When a data packet with $2n$-bit checksum arrives at the receiver, we cannot directly compute the n-bit checksum $\mathbf{c}_n$ associated to the sequence since we do not know the exact values of the $2n$-bit checksum $\mathbf{c}_{2n}$. Instead, we know the probability for each bit of $\mathbf{c}_{2n}$ to be a 1 or a 0, that is what is called the likelihood of $\mathbf{c}_{2n}$ and it is known due to the soft information. So what we can do is to calculate the likelihood ratio of $\mathbf{c}_n$. The problem is : knowing the likelihood ratio of a $2n$-bit checksum computed over a given sequence, what is the likelihood ratio of the n-bit checksum computed over the same sequence?

To answer this question, we have to take a closer look at the relations between the bits of $\mathbf{c}_n$ and the bits of $\mathbf{c}_{2n}$ and for this, we introduce the following notations

- P_i, the probability for the i^{th} bit of $\mathbf{c}_{2n}$ to be a 1, $i \in [0, 2n-1]$. Reciprocally, we note $\overline{P}_i$, the probability for the i^{th} bit of $\mathbf{c}_{2n}$ to be a 0.
- p_i, $i \in [0, n-1]$, the probability for the i^{th} bit of $\mathbf{c}_n$ to be a 1 and $\overline{p}_i$ the probability for the same bit to be a 0.
- q_i, $i \in [0, n-1]$, the probability to have a carry at the i^{th} position of the 1's complement sum between the two mid-parts of $\mathbf{c}_{2n}$. Reciprocally, $\overline{q}_i$ is the probability not to have a carry at the same position.

Considering the *i-th* bit of $\mathbf{c}_n$, we have two possibilities depending on whether there is a carry. If there is a carry, the bit is a 1 if both *i-th* and *i+n-th* bits of $\mathbf{c}_{2n}$ are 1 or both bits are 0. In the other case, if there is no carry from the precedent step, the *i-th* bit of $\mathbf{c}_n$ is a 1 if one of the *i-th* and *i+n-th* bits of $\mathbf{c}_{2n}$ is a 1 and the other is a 0. This gives us the following

equations

$$\begin{cases} p_0 = q_{n-1}.(\overline{P}_0\overline{P}_n + P_0P_n) + \bar{q}_{n-1}(P_0\overline{P}_n + \overline{P}_0P_n) \\ p_i = q_{i-1}(\overline{P}_i\overline{P}_{i+n} + P_iP_{i+n}) \\ \quad +\bar{q}_{i-1}(P_i\overline{P}_{i+n} + \overline{P}_iP_{i+n}), \forall i \in [1, n-1] \end{cases} \tag{4}$$

In (4), the P_i (and thus the $\overline{P}_i$) terms are known since they are contained in the soft information, but the q_i and $\bar{q}_i$ remain unknown for the moment. To determine the value of the *i-th* carry, we have to consider two possibilities: either both *i-th* and *i+n-th* bits of $\mathbf{c}_{2n}$ are 1, or one of those bits is a 1 and the other a 0. In the first case, there is always a carry. In the second case, the *i-th* carry is only a 1 if the *i*-1-*th* carry is a 1 too. This enables us to write the following system of equations

$$\begin{cases} q_0 = q_{n-1}(P_0\overline{P}_n + \overline{P}_0P_n) + P_0P_n \\ q_j = q_{j-1}(P_j\overline{P}_{j+n} + \overline{P}_jP_{j+n}) + P_jP_{j+n}, \forall j \in [1, n-1] \end{cases} \tag{5}$$

The system (5) is made of n equations and has n unknown parameters q_i, $i \in [0, n-1]$. It can be written as

$$\mathbf{Aq} = \mathbf{b} \tag{6}$$

with

$$\begin{cases} \mathbf{q} = (q_0, q_1, ..., q_{n-1})^t \\ \mathbf{b} = (P_0.P_n, P_1.P_{n+1}, ..., P_{n-1}.P_{2n-1})^t \\ \mathbf{A} = \begin{pmatrix} 1 & 0 & \cdots & 0 & -k_0 \\ -k_1 & 1 & \ddots & \ddots & 0 \\ 0 & -k_2 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -k_{n-1} & 1 \end{pmatrix} \end{cases}$$

and

$$k_j = P_j\overline{P}_{j+n} + \overline{P}_jP_{j+n}, \forall j \in [0, n-1]$$

from which $\mathbf{q}$ is easily obtained as

$$\mathbf{q} = \mathbf{A}^{-1}\mathbf{b} \tag{7}$$

Once $\mathbf{q}$ is known, we can use the values q_i, $i \in [0, n-1]$ in (4) in order to obtain p_i, $i \in [0, n-1]$.

3.3. A simplified trellis

Once the likelihood of the n-bit checksum is known, we have to estimate the unknown sequence. As in Section 2.2, we use a trellis structure to do that. The only difference is that the trellis now has 2^n rows instead of 2^{2n}. The number of calculations we have to make with the trellis is almost reduced by a factor 2^n, the complexity is highly reduced. The only calculations we add with this process is the solving of (6) and have a complexity in n^3, which remains negligible compared to the factor 2^n.

4. SIMULATION AND RESULTS

4.1. Context

Our simulation consists in sending packets through an AWGN channel and applying our algorithm to the packets with erroneous headers at the reception. We assume that a soft protocol stack is used at the lower layers, so that we deal with soft information at the IP and UDP-lite layers. The transparent network architecture presented in [5] gives some insights on the way to transmit soft information between protocol layers. Moreover, the UDP-lite layer is set to only protect the header and in both IP and UDP-lite headers, some fields may be *known* or *predictable*. Here are the assumptions we made for those fields

- In UDP-lite, the *Length* field is considered as known since we know the way we use UDP-lite. The *Terminal Port* field is *unknown* but one can assume that a limited number of ports are used at the same time. In our case, it can only take 8 different values.
- In IP, The *Version* field is *known* since we assume we use IPv4. The *Header Size* field is considered as *known*. The *Type of Service* field is also assumed to be *known*. The *Length* field is considered as *predictable* as it can be obtained from the MAC layer. The first bit of the *Flags* field is reserved and set to 0, so it is *known*. The *Protocol* field is *known* as UDP is known to be the upper layer.

The remaining fields are all considered as *unknown*.

4.2. Results

At the receiver, we compare the HER (Header Error Rate) of the received packets for different values of the SNR. We compare three different decoders: the standard one which decodes the bits of the header sequence according to their likelihood ratio, our header-correcter decoder based on the redundancy included in the 16-bit checksum and the same decoder with reduced complexity as we explain in Section 3.2.

4.2.1. IP header correction

Figure 1 shows this comparison for the IP header. As expected, both proposed decoders show better results than the standard one. The one which bases its header correction on the 16-bit checksum always have the fewer HER, whereas the one with reduced complexity is always between the two other decoders in term of results. We explain this by the fact that when we work with the reduced complexity version, we turn the likelihood ratio of the real 16-bit checksum into a likelihood ratio of a corresponding 8-bit checksum. But this operation is not reversible, when we do this, we lose redundancy about the unknown sequence.

If we have a closer look at the uncorrected error patterns that remain after we tried to correct them, we clearly notice

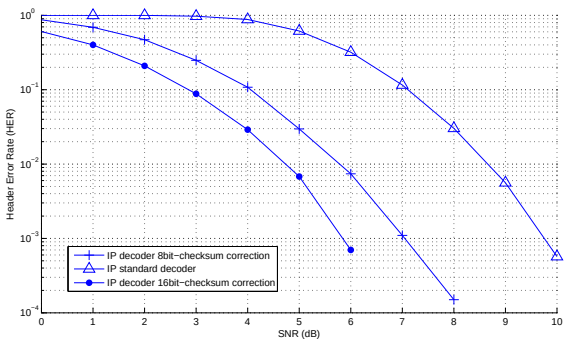

Fig. 1. HER vs SNR for the standard, 16-bit header correcter and 8-bit header correcter decoders of the IP layer.

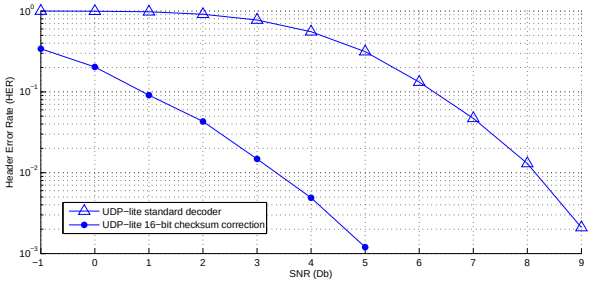

Fig. 2. HER vs SNR for the standard and 16-bit header correcter decoders of the UDP-lite layer

that there is no single bit error in the corrected sequence. This due to the checksum property of always detecting single bit error. The most common type of error appears when two bits separated by 15 (or 7 for the reduced complexity version) bits are both wrong but when added to each other, they do not change the checksum value. This kind of error is not detectable by our algorithm.

4.2.2. UDP-lite header correction

Figure 2 compares the performance of the standard decoder and the basic header-correcter decoder for the UDP-lite layer. Once again, our decoder has better results than the standard one and if we compare it with the same decoder at the IP layer, we notice that the results are even better. For a HER of 10^{-2}, our UDP decoder needs 3.33 dB whereas the IP decoder needs 4.75 dB. We explain this observation by two facts. First, the length of the unknown sequence in the UDP-lite header is smaller that the unknown sequence of the IP header. It is 32 bits instead of 39, there are fewer chances for errors to occur. Second, in the UDP-lite header, the *Terminal Port* field can only take a given number of values which correspond to the opened ports of the Terminal, which highly reduces the possible values for this field.

5. CONCLUSION AND PERSPECTIVE

By using the redundancy present in the checksum, the solution we propose enables us to make the upper protocol layers permeable to corrupted packets.

Currently, the main drawback of this method is its calculation complexity. If it is longer to estimate the correct header of a corrupted packet rather than waiting for it to be correctly sent, our contribution is useless. The solution we propose is fine as it highly reduces the number of calculations needed for the method but it is not as efficient in term of errors correction. To continue this project, the main purpose is to keep on working about complexity reduction. Instead of replacing the 16-bit checksum by the corresponding 8-bit checksum, it would be interesting to keep the 16-bit checksum and try to estimate the unknown sequence with two trellis of 2^8 rows, one corresponding to the 8 most significant bits of the 16-bit checksum and the other corresponding to 8 least significant bits. Of course, they would not be independent trellis, they should share information about their respective carries. This way, we would still have a reduced complexity but our sequence estimation should be better.

6. REFERENCES

[1] R. E. Blahut. *Theory and Practice of Error Control Codes*. Addison-Wesley, Reading, MA, 1984.

[2] R. T. Braden, D. A. Borman, and C. Partridge. RFC 1071: Computing the Internet checksum, September 1988. Updated by RFC1141.

[3] Duhamel and Kieffer. *Joint Source-channel Decoding: A Cross-layer Perspective With Applications in Video Broadcasting*. 2009.

[4] J. F. Kurose and K. W. Ross. *Computer Networking: A Top-Down Approach Featuring the Internet*. Addison Wesley, Boston, third edition, 2005.

[5] C. Lamy-Bergot J. Huusko M. G. Martini, M. Mazzoti and P. Amon. Content adaptive network aware joint optimization of wireless video transmission. *IEEE Communications Magazine*, 45(1):8490, 2007.

[6] J. K. Wolf. Efficient maximum-likelihood decoding of linear block codes using a trellis. *IEEE Trans. Inform. Theory*, 24(1):76–80, 1978.